Stochastic feedback-based control of QoS in soft real-time systems

T. Cucinotta, L. Palopoli, L. Marzario

***Abstract*— This paper investigates application of feedback based control mechanisms to the problem of scheduling soft real-time tasks, so to meet certain quality of service (QoS) requirements. First, a stochastic model is introduced for a task evolving under the effect of a feedback based controller, where the uncertainties due to the apriori unknown execution times of the jobs are catched in terms of an input stochastic process. The problem of control is formalised in the stochastic domain, by expressing QoS requirements in terms of stochastic properties to be satisfied by the system state evolution process. Control laws satisfying some of the stated requirements are introduced, and fundamental facts are proved on the closed loop system dynamics under the effect of such controllers, such as stochastic stability. Finally, experimental results are presented gathered by an implementation of the controllers in the Linux kernel, showing feasibility and effectiveness of the proposed approach in controlling the QoS experienced during a video decoding application.**

I. Introduction

The ubiquitous presence of networked computing systems in consumer electronics has emphasized the importance of appropriate resource allocation policies: i.e., a limited pool of computing elements and resources have to be shared between different contending requests so that each user receives a specified level of Quality of Service (QoS). This problem offers important opportunities for the application of feedback control theory. Generally speaking, the amount of resource necessary to sustain a specified QoS level for an application changes in time and a feeback based adaptation can make the resource management more efficient provided that an appropriate set of "sensors" be available to collect measurements.

This type of applications has gained momentum in the last few years with particular focus on the control of computer networks and queues [1]. An emerging trend is the application of feedback control techniques to the management of shared resources inside the operating system of a computer. In a modern computer system, even the simplest one, it frequently occurs that multiple software programs (tasks) run concurrently competing for the use of CPU, RAM memory, disks, etc.... The operating system is the component in charge of scheduling the access of tasks to the shared resources. This activity becomes particularly challenging when the applications implemented by software tasks are time-sensitive, as it is the case of a multimedia application. In this case schedulers have to allocate the machine resources so that real-time constraints are met. For many applications real-time constraints are *soft*, i.e., occasional violations entail only QoS degradations without any danger for the integrity of the system or for the people safety. As an example, for a video streaming application it is not necessary to decode every frame within a fixed interval as long as fluctuations in the decoding rate do not overcome the threshold of human perception. In this context properly designed schedulers offer considerably better performance than conventional solutions. The most important feature required to these scheduling algorithms is to approximate a "fluid" allocation of the resources such as Resource Reservation (RR) [2] and Proportional Share [3]: the scheduler can allocate a fraction (bandwidth) of the shared resources to each task (in the sequel we will restrict to the problem of CPU allocation). This technology is not by itself sufficient to solve the problem of QoS driven CPU allocation to tasks: how should one choose the bandwitdth allocated to each task ? This choice is dictated by the execution requirements of each task that are hardly known beforehand and may dramatically change in time. This is a strong motivation for the application of feedback control to adjust the bandwidth based on QoS measures. It is noteworthy that for a feeback solution to be acceptable in this context, its introduced overhead must be negligible as compared to the normal bookkeeping activities of the scheduler. A first proposal of this kind for time sharing systems dates back to 1962 [4]. More recently, feedback control techniques have been applied to real-time scheduling [5], [6], [7], [8] and multimedia systems [9], [10]. Owing to the difficulties in modelling schedulers as dynamic systems, these works could offer little analytical evidence of the effectiveness of their approaches. The use of scheduling algorithms such as the RR allows us to fill in this gap. Based on a precise dynamic model for the system's evolution, a switching PI strategy was developed [11] and formal proofs on the stability of the resulting schemes were offered in [12]. In [13] ad-hoc nonlinear feedback laws were proposed for the system. The sequence of execution times for a task plays the role of a noise term, which is characterised in terms of its worst case effects on the system stability. A significant drawback with this approach is that any knowledge on the stochastic properties of the execution times process is dissipated. On the other hand, it frequently occurs that such a knowledge is available (e.g., the decoding time of a MPEG player) and that it can be leveraged to improve the system's performance. To achieve this goal, the control design problem is best attacked in the stochastic domain.

The first goal of this paper is to show possible feedback design laws based on the theory of stochastic control. The

second goal is to show a stability criterion that can be applied to assess the stability of the closed loop system in a stochastic sense, i.e., the existence and uniqueness of a steady state invariant probability distribution for the controlled quantity. The technological feasibility of the proposed approach by means of a minimally invasive set of modifications to the Linux Operating System is shown in [14].

II. Problem description

In this section we describe the problem of QoS control in soft real-time systems. Before getting into the presentation of the problem, we need to quickly introduce some basic concepts and terminology concerning real-time scheduling.

A. Basic definitions on real-time processor scheduling

A real-time computing system consists of a set of concurrently running software activities (tasks) $\{\tau^{(i)}, i = 1, .., n\}$. Each task $\tau^{(i)}$ is a program consisting of a stream of jobs (execution instances) $\{J_k^{(i)}\}_{k=1,2,...}$. The k^{th} job $J_k^{(i)}$ of the i^{th} task arrives (i.e. becomes executable) at time $r_k^{(i)}$ and requires a variable computation time $e_k^{(i)}$; the finishing time of $J_k^{(i)}$, depending on the scheduling performed by the Operative System (O.S.), is denoted as $f_k^{(i)}$. For real-time taks, each job $J_k^{(i)}$ is associated a deadline $d_k^{(i)}$, which is said to be *respected* if $f_k^{(i)} \leq d_k^{(i)}$, *violated* otherwise ($f_k^{(i)} > d_k^{(i)}$). While in the hard real-time setting all deadlines must be respected, for soft real-time tasks occasional deadline misses can be tolerated provided that this deviation be kept in check. We will be more formal on this issue later. For the scope of this paper, we will consider only periodically activated tasks, i.e. $r_k^{(i)} = kT^{(i)}$, where $T^{(i)}$ is the task's activation period. Moreover, for the sake of simplicity, the deadline of a job will be set equal to the activation instant of the next one: $d_k^{(i)} = r_{k+1}^{(i)}$.

A quantity of interest for some real-time applications based on periodic tasks is the so called *jitter* on the finishing time, defined as the difference $F_k^{(i)} = f_k^{(i)} - f_{k-1}^{(i)} - T^{(i)}$. A small value of the jitter allows one to consider the task as a fixed delay element and to use this information in the design of the overall system.

B. Resource Reservation scheduling

As multiple tasks run on the same CPU, a scheduling algorithm is needed to mediate contending requests of execution. The present work is focused on the analysis and control of a set of tasks scheduled according to a class of algorithms known as Resource Reservation (RR), first proposed in [2] then developed in [15], [16], [17], [18]. Using such techniques, each task $\tau^{(i)}$ is associated a pair $[Q^{(i)}, P^{(i)}]$ meaning that the task is guaranteed a *budget* of $Q^{(i)}$ execution time units for every allocation period $P^{(i)}$, whenever in need. Thus the ratio $b^{(i)} \triangleq \frac{Q^{(i)}}{P^{(i)}}$ is the utilisation fraction of the CPU allocated to the task. The allocation period $P^{(i)}$ may be arbitrarily chosen in RR techniques; for reasons explained below we will assume small values of $P^{(i)}$ as compared to the task period $T^{(i)}$. Under this assumption it is possible to show that RR scheduling approximates a fluid allocation of the processor, i.e. one where each task executes as if on a dedicated slower processor having speed equal to a fraction (equal to $b^{(i)}$) of the real CPU. In order to illustrate this concept, define the *virtual finishing time* $v_k^{(i)}$ as the time a job $J_k^{(i)}$ would finish if it were running on a dedicated processor of speed $b^{(i)}$ times the real CPU speed. It is possible to prove [17] that, if the RR paradigm is strictly applied (i.e. a task $\tau^{(i)}$ receives exactly $Q^{(i)}$ ticks every $P^{(i)}$ even when more processor time could be available), then the relation

$$v_k^{(i)} - \delta^{(i)} \leq f_k^{(i)} \leq v_k^{(i)} + \delta^{(i)} \tag{1}$$

holds, where $\delta^{(i)} \triangleq (1 - b^{(i)})P^{(i)}$, showing that a resource reservation scheduler approximates a fluid allocation up to a granularity dictated by the choice of $P^{(i)}$. Based on this result, the dynamic model used for control design in this paper assumes a fluid allocation. Furthermore, this abstraction allows us to apply the results shown in this paper also to other classes of scheduling algorithms that approximate a fluid model, such as Proportional Share (PS) [3]. Note also that, from a practical standpoint, $P^{(i)}$ cannot be chosen too small, because of the higher overhead introduced for the frequent context switches between the tasks.

Clearly, whenever multiple tasks populate a processor, the total sum of the bandwidths $b^{(i)}$ cannot exceed the processor's availability of computing power: $\sum_i b^{(i)} \leq 1$. As a matter of fact, depending on the efficiency of the scheduling algorithm, the total availability of bandwidth U may in general be lower than 1 :$\sum_i b^{(i)} \leq U \leq 1$.

C. Quality of Service metrics and dynamic model

In order to properly adjust the scheduling parameters (e.g. $Q^{(i)}$ and $P^{(i)}$) of each task, it is of paramount importance to quantify the Quality of Service (QoS) that the task experiences during its execution. Since in our model we tolerate occasional violation of the deadlines, it is reasonable to use, as QoS metrics, the *scheduling error* (s.e.) defined as the difference between the finishing time of a job and its deadline, measured relatively to the task period. Referring to the fluid allocation assumption, the definition of this quantity is $\varepsilon_k^{(i)} = \frac{v_k^{(i)} - d_k^{(i)}}{T^{(i)}}$. An ideal bandwidth allocation would be one for which $\varepsilon_k^{(i)} = 0$ for all k and i. Indeed, both $\varepsilon_k^{(i)} > 0$ and $\varepsilon_k^{(i)} < 0$ are undesirable situations, since in the former case the job does not respect its timing constraint, whilst in the latter one it is given more bandwidth than strictly required, stealing resources to other possible activities into the system. Concerning the problem of controlling jitter fluctuations, instead, it is easy to see that, if the evolution of $\varepsilon_k^{(i)}$ is constrained in *a* small interval, then also the jitter is bounded in a small interval.

For notational convenience, the task superscript (i) will be dropped from now on whenever all cited quantities

refer to the same task. Furthermore, the symbol $c_k$will be used as a shortand for e_k/T. The dynamics of ε_k will be characterised by a function $\mathfrak{S}_C(\cdot)$, highlighting dependence of the s.e. value at the next step from the s.e. at the current step, the allocated bandwidth for the next job execution, and its actual computation time. This may be formalised as [19]

$$\varepsilon_{k+1} = \mathfrak{S}_C(\varepsilon_k, c_k, b_k) = s(\varepsilon_k) + \frac{c_k}{b_k} - 1, \qquad (2)$$

where $s(\cdot)$ is defined as $s(x) \triangleq x$ if $x > 0$ and $s(x) \triangleq 0$ otherwise.

D. Stochastic model

In this subsection a stochastic evolution model is introduced for the system under consideration, where all time-varying quantities of interest introduced so far are modeled as discrete-time stochastic processes (s.p.), with the time instants at which a task evolution is observed being equal to the job finishing times $\{f_k\}$. Thus, the following notation will be used in the remind of this section: given a discrete-time stochastic process $\{X_k\}$, $f_{X_k}(x)$ and $F_{X_k}(x)$ will denote, respectively, the probability density function (p.d.f.) and the cumulative density function (c.d.f.) for the stochastic variable (s.v.) X_k; μ_{X_k} and σ_{X_k} will denote, respectively, the expected value and the standard deviation of X_k. Finally, $Pr\{\cdot\}$ will denote the probability operator, which applied to an event outputs the probability that the event occurs, and $E[\cdot]$ denotes the expectation operator, which applied to a s.v. outputs its expected value.

For each task, given the impossibility of exact predictive knowledge of the job execution times, the sequence $\{c_k\}$ is modeled as the discrete-time, continuous-state s.p. $\{C_k\}$, where the k^{th} job execution time is modeled as the s.v. C_k. The control action can be modeled by a function $\mathfrak{B}(\cdot)$ that, at each step k, assigns the bandwidth to be reserved for the next job depending on the current system state and the sequence of past job execution times. Any case, the assigned value cannot be greater than a prefixed limit $B_H \in]0, 1]$ that is allocated on a per-task basis:

$$b_k = \mathfrak{B}(\varepsilon_k, B_H, \{c_{k-1}, c_{k-2}, \dots\}). \qquad (3)$$

Defining the initial scheduling error as $\varepsilon_1 \triangleq 0$, and considering the dynamic evolution in Equation 3 and 2, the sequence of bandwidth and scheduling error values are modeled as stochastic processes as well, respectively denoted as $\{B_k\}$ and $\{\mathcal{E}_k\}$. Evolution of the s.e. process is stated in terms of the relation between stochastic variables given by the $\mathfrak{S}_C$ function introduced above:

$$\mathcal{E}_{k+1} = \mathfrak{S}_C(\mathcal{E}_k, B_k, C_k) = s(\mathcal{E}_k) + \frac{C_k}{B_k} - 1. \qquad (4)$$

The stochastic properties of the process $\{C_k\}$ are application specific, depending on the type of required computation activities. For example, an MPEG decoder is modelable by a s.p. with a different type and parameters than a control application. Thus, feedback mechanism to use is to be calibrated on specific classes of applications. A simple case of interest that will be considered later for the purpose of stochastic staiblity is the one where the stochastic variables $\{C_k\}$ are independent and identically distributed (i.i.d.).

E. Stochastic evalutation of control performance

As far as the current work is concerned, quality of service is defined in terms of the scheduling error experienced by a task. Given the representation of the s.e. evolution as a s.p., the QoS experienced by a task can be defined in terms of the stochastic properties of that process. Specifically, the first order probability density function $f_{\mathcal{E}_k}(\cdot)$ may be used to make qualitative comparisons among different control algorithms, by plotting the resulting distributions on the same graph. A precise assesment of the effectiveness of an algorithm in controlling the s.e. evolution may be done on the basis of the achieved s.e. expected value $\mu_{\mathcal{E}_k} = E[\mathcal{E}_k]$ and standard deviation $\sigma_{\mathcal{E}_k} = E\left[(\mathcal{E}_k - \mu_{\mathcal{E}_k})^2\right]$. Another metrics of interest can be defined in terms of the probability that the s.e. resides within a prefixed interval I near the origin $\Pr\{\mathcal{E}_k \in I\}$. Such metrics, generally dependent on the time subscript k, will assume particular relevance for $k \to \infty$ whenever the resulting process reaches stochastic equilibrium, representing the expected behaviour of the closed loop controlled system in the long run.

III. Control schemes

This section explores control schemes dictated by different requirements on the closed loop system behaviour. The primary purpose of each controller is ensuring that the experienced QoS does not degrade below acceptable values; in terms of the scheduling error this means that ε_k should never be too positive (meaning a severe violation of the deadline). In addition, we aim at an efficient utilisation of the CPU: the bandwidth b_k allocated to each job should be minimal to make room for other applications. The different control schemes presented below achieve different tradeoffs between these two requirements. Their behaviour is specified in terms of the stochastic properties of the system's expected evolution for the next step, given current state and the predicted behaviour of the process C_k. The structure of the resulting controllers is comprised of two blocks (see Figure 1): a block predicting information on the evolution of C_k (predictor) and a block implementing a feedback law based on the measured value of ε_k and on the information received from the predictor.

Generally speaking, the stochastic requirements will be formulated below as conditoned probabilities and expectations, where the conditioning event is the knowledge of the system past evolution that the controller uses. In particular, if the process $\{C_k\}$ is correlated, a good prediction is based on the knowledge of the past samples of $\{C_k\}$. In order to formalise this concept, we first introduce the vector z_k of past computation times occurred during a task evolution up to a state k as $z_k \triangleq [c_{k-1}, c_{k-2}, \dots]$, and its stochastic counterpart $Z_k \triangleq [C_{k-1}, C_{k-2}, \dots]$. Then, we observe that, in case of correlated $\{C_k\}$, the expected behaviour of the input process, as well as the expected QoS achieved

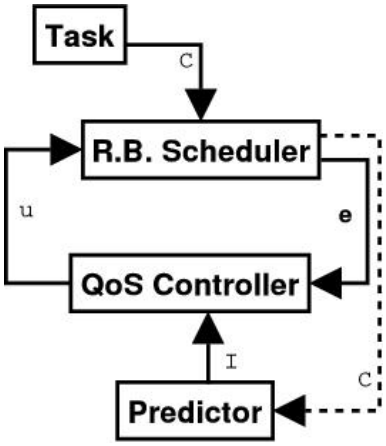

Fig. 1. Control loop. Assigned bandwidth depends on the current s.e. value and on the predicted behaviour of the input process.

at the next step, are not only conditioned by the current s.e. value ε_k, but also (potentially) by the entire input process history represented by z_k, and known (in principle) to the controller. This is translated in conditioning all involved probability and expectation operators used below, for the purpose of stating requirements and properties of the various controllers, not simply to the $\{\mathcal{E}_k = \varepsilon_k\}$ event, but to the joint $\{\mathcal{E}_k = \varepsilon_k \wedge Z_k = z_k\}$ event. Note that, in the simple case of independent $\{C_k\}$ process, the past history z_k does not carry any valuable information. For notational simplicity and without loss of generality, in the discussion below, we will refer to this simpler case.

Clearly, conditioned stochastic properties ensured by the different control schemes are not generally related to the unconditioned properties. The latter can only be assessed on the invariant steady state distribution, if it exists. This topic will be discussed in the next section.

A common requirement for all control schemes – implicitly assumed in the discussion below – is that the bandwidth chosen by the controller cannot exceed a maximum value B_H. Furthermore, all control schemes have been designed with complexity in mind: the computations that take place within the OS before a scheduling decision need to be reduced to the bare minimum.

A. Stochastic invariant set

In the first control scheme, it is assumed that, at step k, the predictor provides an interval $[h_k, H_k]$ where C_k is expected to fall with a high probability. The control goal is constraining the evolution of the s.e. within a fixed interval, called *invariant set* $\mathcal{I} \triangleq [-e, E]$, with $e \in [0, 1[$, $E \in [0, +\infty[$, whenever the input process respects the prediction. It is possible for the s.e. to occasionally evolve outside of the invariant set, due to wrong predictions. Different tradeoffs can be achieved by different choices of e and E. In particular, if the only concern is the probability of a deadline miss, one can simply set $\mathcal{I} = [-1, E]$, whereas a *narrow* $\mathcal{I}$ corresponds to requiring small jitter fluctuations so that the assigned bandwidth remains always close to the strictly necessary C_k. In stochastic terms, this goal can be formulated as follows.

Requirement 1: Given a s.e. evolution range $\mathcal{I}$ and a probability p, at each step k with $\mathcal{E}_k = \varepsilon_k \in \mathcal{I}$, guarantee that $Pr\{\mathcal{E}_{k+1} \in \mathcal{I} \mid \mathcal{E}_k = \varepsilon_k\} \geq p$.

The following result provides a family of control laws meeting this requirement.

Proposition 1: If, at each step k, an interval $[h_k, H_k]$ is known to the controller, such that $\Pr\{C_k \in [h_k, H_k]\} \geq p$, then Requirement 1 is satisfied by any controller choosing a bandwidth value in the range $\mathfrak{B}(\varepsilon) \in$

$$\left[\frac{H_k}{1 + E - s(\varepsilon_k)},\ min\left\{\frac{h_k}{1 - e - s(\varepsilon_k)}, B_H\right\}\right], \qquad (5)$$

under the assumption that $e + \frac{h_k}{H_k} E \geq 1 - \frac{h_k}{H_k}$ and $B_H \geq H_k$.

Proof: $Pr\{\mathcal{E}_{k+1} \in [-e, E] \mid \mathcal{E}_k = \varepsilon_k\} = Pr\{b_k[1 - e - s(\varepsilon_k)] \leq C_k \leq b_k[1 + E + s(\varepsilon_k)]\}$. Choosing b_k in the range dictated by Equation 5, which is guaranteed to be not empty under the stated assumptions (see [13, Theorem 1]), implies $b_k[1 - e - s(\varepsilon_k)] \leq h_k$ and $b_k[1 + E + s(\varepsilon_k)] \geq H_k$, thus: $Pr\{b_k[1 - e - s(\varepsilon_k)] \leq C_k \leq b_k[1 + E + s(\varepsilon_k)]\} \geq Pr\{h_k \leq C_k \leq H_k\} \geq p$. ■

Clearly, with the introduced controller, a *recovery policy* must be undertaken when the state falls outside of the invariant set $\mathcal{I} = [-e, E]$. A reasonable policy is a minimum time strategy; it corresponds to choosing, for $\varepsilon_k > E$, the maximum available bandwidth that preserves $\varepsilon_{k+1} \geq -e$ whenever $C_k \geq h_k$. This can be formulated as: $\mathfrak{B}(\varepsilon_k) =$

$$\begin{cases} min\left\{B_H, \frac{h_k}{1 - e - \varepsilon_k}\right\} & if\ E < \varepsilon_k < 1 - e \\ B_H & if\ \varepsilon_k \geq 1 - e \end{cases} \qquad (6)$$

The just introduced control law is equal to the one introduced in the context of deterministic systems in [13], where also necessary and sufficient conditions for its existence are stated, along with a detailed discussion of its properties.

The prediction algorithm for constructing the interval $[h_k, H_k]$ is application specific. As an example, for a video decoder, it could be possible to perform a quick estimation of the upper and lower bounds for a frame decoding time by looking at the frame structure before starting the decoding itself. For independent, identically distributed (i.i.d.) C_k, such interval does not depend on k.

B. Stochastic dead-beat

The invariant-based technique presented above provides the user with a relatively fine control of the system performance by different choices of the interval extremes. However, this freedom for some applications can also be a source of unrequired complexity in the parameter tuning and in the design of the predictor. In some cases, an effective goal can just be limited to the control of the next s.e. expected value, as formalised in the following.

Requirement 2: Fixed a *target* s.e. value ε^*, guarantee that $E[\mathcal{E}_{k+1} \mid \mathcal{E}_k = \varepsilon_k] = \varepsilon^*$.

The following yields a control law satisfying such a requirement.

Proposition 2: If, at each step k, the expected value μ_{C_k} of the next job execution time C_k is known, then Requirement 2 is satisfied by a controller setting $\mathfrak{B}(\varepsilon_k) =$

$$\begin{cases} \frac{\mu_{C_k}}{1-\varepsilon^*-s(\varepsilon_k)} & if\ \varepsilon_k < 1 - \frac{\mu_{C_k}}{B_H} \\ B_H & if\ \varepsilon_k \geq 1 - \frac{\mu_{C_k}}{B_H} \end{cases}. \quad (7)$$

Proof: The relation 4 directly implies $E[\mathcal{E}_{k+1} \mid \mathcal{E}_k = \varepsilon_k] = s(\varepsilon_k) + \frac{E[C_k]}{\mathfrak{B}(\varepsilon_k)} - 1$. By imposing the expected next s.e. value to be equal to the target value ε^*, it is easily obtained

$$\mathfrak{B}(\varepsilon_k) = \frac{\mu_{C_k}}{1-\varepsilon^*-s(\varepsilon_k)}.$$

Considering the upper bound constraint on the set of possible bandwidth values, the proposition statement easily follows. ■

A particular case of interest is obtained by setting $\varepsilon^* = 0$ in Equation 7, achieving, at each step, a null expected scheduling error for the next job. As compared to the invariant based controller, the SDB requires a looser knowledge of the input stochastic process, i.e. just the μ_{C_k} parameter.

C. Optimal cost

One of the most effective way of trading performance (QoS) vs command cost (bandwidth) is by formulating the control problem as an optimal control problem. Taking inspiration from the dynamic programming techniques [1, pgg. 536-542], it is possible to define, for each state transition, a *cost function*, depending on the system state at the next step and the bandwidth allocated at the current step $W(\varepsilon_{k+1}, b_k)$. Then, the control law chooses the bandwidth so as to optimise the cost function.

As an example, we have analysed a cost function that depends linearly from the assigned bandwidth value, and quadratically from the achieved value of the s.e. at the next step, where a weight factor $\gamma \in]0,1[$ is used for tuning the importance given to the s.e. deviation with respect to the one given to assigning high bandwidth values. Such cost optimal requirement on the stochastic evolution of the system may be formalised as follows.

Requirement 3: Given the cost function $W(\varepsilon_{k+1}, b_k) = \gamma\varepsilon_{k+1}^2 + (1-\gamma)b_k$, with $\gamma \in]0,1[$, associating, at each step k, a cost to the next system transition, guarantee that the expected value of such cost $E\left[W(\mathcal{E}_{k+1}, b) \mid \mathcal{E}_k = \varepsilon\right]$, conditioned to the current system state, be minimum.

This problem may be solved in closed form, as reported in the following result.

Proposition 3: If the mean value μ_C and the standard deviation σ_C of the next job execution time C_k are known, the bandwidth assignment $\mathfrak{B}(\varepsilon)$ satisfying Requirement 3, subject to the constraint $\mathfrak{B}(\varepsilon) \leq B_H$, is given by the formula

$$\begin{aligned} B(\varepsilon) &= \min\left\{ \sqrt[3]{\rho + \delta(\varepsilon)} + \sqrt[3]{\rho - \delta(\varepsilon)}, B_H \right\} \\ \rho &\triangleq \frac{\gamma(\sigma_C^2 + \mu_C^2)}{(1-\gamma)} \\ \delta(\varepsilon) &\triangleq \sqrt{\left[\frac{\gamma(\sigma_C^2 + \mu_C^2)}{1-\gamma}\right]^2 + \left(\frac{2}{3}\frac{\mu_C\gamma}{1-\gamma}\left[1 - s(\varepsilon)\right]\right)^3}. \end{aligned}$$

Proof: The expected total cost (conditioned to $\mathcal{E}_k = \varepsilon$, omitted for notational convenience), under assignment of $b_k = b$, results: $E\left[W(\mathcal{E}_{k+1}, b)\right] =$

$$\begin{aligned} &= \gamma E\left[\left(s(\varepsilon) + \frac{C_k}{b} - 1\right)^2\right] + (1-\gamma)b = \\ &= \gamma\left\{[s(\varepsilon)-1]^2 + \frac{\sigma_C^2 + \mu_C^2}{b^2} + 2\frac{\mu_C[s(\varepsilon)-1]}{b}\right\} + (1-\gamma)b \end{aligned}$$

Searching the minimum of $E\left[W(\mathcal{E}_{k+1}, b)\right]$ leads to:

$$\frac{\partial}{\partial b}E\left[W(\varepsilon, b)\right] = 2\gamma\left\{\frac{\mu_C[1-s(\varepsilon)]}{b^2} - \frac{\sigma_C^2 + \mu_C^2}{b^3}\right\} + 1 - \gamma = 0$$

$$b^3 - \frac{2\gamma\mu_C[s(\varepsilon)-1]}{1-\gamma}b - \frac{2\gamma(\sigma_C^2 + \mu_C^2)}{1-\gamma} = 0$$

The final solution is obtained using the well known formula for polynomials of third degree. It is also possible to verify that this value of b corresponds to a minimum of the expected cost $E\left[W(\varepsilon, b)\right]$. Proof of this fact can be found in [20] and is omitted for the sake of brevity. ■

Remark 1: The above shown formula can also be used in the case of imaginary $\delta(\varepsilon)$, by making computations in the complex domain, and with a proper computation of the two cubic roots, so that in the final sum the two imaginary parts cancel each other.

The controller based on this approach is the most flexible among the presented schemes: different tradeoffs between QoS and the applied bandwidth are simply decided by choosing a value for γ. However, required computations are quite involved and their viability inside the OS is still under evaluation.

IV. Stochastic stability of the proposed controllers

When evalutating correctness of the QoS control strategies introduced above, two properties are of great interest: stochastic stability, i.e., the existence and uniqueness of an invariant probability function for $\{\mathcal{E}_k\}$ when $k \to \infty$, and ergodicity of the resulting process. In the following, we focus on the simple case of i.i.d. job execution times. Extensibility of some of the exposed concepts to the more general case of not i.i.d. input process, especially in those cases in which the application domain allows the correlated input process to be tightly modeled as an independent process filtered through a linear system, is still work in progress and needs further investigation.

This section, after expliciting the Markov operator associated to the system evolution, provides two general criteria stating sufficient conditions for the existence and uniqueness of an invariant p.d.f. for the controlled system. Such criteria are directly applicable to all of the controllers introduced so far. Then, some properties of the stochastic equilibrium are stated as well.

A. Basic definitions and Markov operator

In this section, we will use terminology and results taken from [21], comprising the concepts of Markov Chains (MC) as defined in section 2.2, and weak-Feller property of a MC, as defined in (2.2.2). Such definitions are not reported here for the sake of brevity.

Let $f_k(\cdot)$ represent the p.d.f. of the scheduling error at step k, and let $f_0(\cdot)$ be the scheduling error p.d.f. at step 0. The system may be associated a Markov operator $\mathcal{P} : D \to D$, where D denotes the set of probability density functions on the $[-1, +\infty[$ range, such that $f_{k+1} = \mathcal{P} f_k$. When the system is controlled by an appropriate (this will be discussed more deeply later) law $\mathfrak{B}(\cdot)$, it is expected that the function sequence $\{f_k\}$ converges to the invariant p.d.f. $f = \lim_{k\to+\infty} f_k$, independent from $f_0(\cdot)$, and possessing the fundamental property of being the fixed point of the $\mathcal{P}$ operator, i.e. $f = \mathcal{P} f$.

The Markov operator for our system can be easily explicited by considering the probability $f_{k+1}(y)\,dy = Pr\{y \le \mathcal{E}_{k+1} < y + dy\}$, the relation 4 between the stochastic variables $\mathcal{E}_k$, C_k,$\mathcal{E}_{k+1}$, and conditioning to all possible values of the scheduling error at the current step, leading to

$$\mathcal{P} f(y) = \int_{-1}^{+\infty} f_{C_k}\left[\mathfrak{B}(x)(1 + y - s(x))\right] \mathfrak{B}(x) f(x) dx. \tag{8}$$

Another quantity of interest, when dealing with properties of our system at stochastic equilibrium, is the *stochastic kernel* associated to the operator $\mathcal{P}$, i.e. the function $K(x, y)$ such that: $\mathcal{P} f(x) = \int K(x, y) f(y) dy$. From the explicit representation of $\mathcal{P}$ of Equation 8, it easily follows

$$K(x, y) \triangleq f_C\left[\mathfrak{B}(y)(1 + x - s(y))\right] \mathfrak{B}(y). \tag{9}$$

B. Sufficient conditions for the existence and unicity of an invariant p.d.f.

Theorem 1: Consider the system defined by Equation 4, evolving under the action of an input process $\{C_k\}$ i.i.d., for which the mean value and standard deviation exist, under the control of a generic control function $\mathfrak{B}(\cdot)$ satisfying the following requirements:
(b_1) $\mathfrak{B}(\cdot)$ is continuous
(b_2) $\mathfrak{B}(\cdot)$ is upper bounded by $B_H > \mu_C$, and $\forall \varepsilon \ge \varepsilon_H$, $\mathfrak{B}(\varepsilon) = B_H$
(b_3) $\mathfrak{B}(\cdot)$ is lower bounded by a $B_L > 0$.
Then, an invariant p.d.f. exists for the scheduling error evolution.

Proof: The proof is given applying theorem 7.2.4 in [21]. This theorem states that sufficient conditions for the existence of an invariant probability function are: (a) the MC respects the weak-Feller property, and (b) fixed an arbitrary, continuous, strictly positive function $f_0(\cdot)$, with the property that $\lim_{x\to\infty} f_0(x) = 0$, a nonnegative number b and a nonnegative measurable function $V(\cdot)$ not identically null exist, s.t. $\forall x$, $E[\mathcal{E}_{k+1} \mid \mathcal{E}_k = x] \le V(x) - 1 + b f_0(x)$.

(a). As shown in the note at page 58 of [21], a MC of the form $\mathcal{E}_{k+1} = F(\mathcal{E}_k, C_k)$, with $\{C_k\}$ i.i.d., is weak-Feller if $F(\cdot, y)$ is continuous $\forall y$. In our case, $F(x, y) \equiv s(x) + \frac{y}{\mathfrak{B}(x)} - 1$ is always continuous, under the assumptions that the $\mathfrak{B}(\cdot)$ function be continuous and strictly positive, as from the assumptions.

(b). Define V as $V(x) \triangleq x^2$. Then, $E[V(\mathcal{E}_{k+1}) \mid \mathcal{E}_k = x] = E\left[(s(x) + \frac{C_k}{\mathfrak{B}(x)} - 1)^2\right] = [s(x) - 1]^2 + \frac{\mu_C^2 + \sigma_C^2}{\mathfrak{B}^2(x)} + 2[s(x) - 1]\frac{\mu_C}{\mathfrak{B}(x)} \le x^2 - 1 + b f_0(x)$, where last inequality must be verified. For positive values of x, we have: $2(1 - x)\left(1 - \frac{\mu_C}{\mathfrak{B}(x)}\right) + \frac{\mu_C^2 + \sigma_C^2}{\mathfrak{B}^2(x)} \le b f_0(x)$. Consider the value x^* defined as $x^* \triangleq max\{\varepsilon_H, 1\}$. It is clear that, if $\mu_C < B_H$, $\forall x > x^*$, the left member becomes negative, and diverges to $-\infty$ as $x \to \infty$. Thus, due to the strictly positive lower bound $B_L > 0$ on $\mathfrak{B}(x)$, a b value satisfying the inequality always exists: $b \ge \max_{x\in[-1,x^*]}\left\{\frac{1}{f_0(x)}\left[2(1 - x)\left(1 - \frac{\mu_C}{\mathfrak{B}(x)}\right) + \frac{\mu_C^2 + \sigma_C^2}{\mathfrak{B}^2(x)}\right]\right\}$. For negative values of x, instead, we have: $2\left(1 - \frac{\mu_C}{\mathfrak{B}(x)}\right) + \frac{\mu_C^2 + \sigma_C^2}{\mathfrak{B}^2(x)} \le x^2 + b f_0(x)$. As $x \in [-1, 0]$, it is sufficient to take: $b \ge \max_{x\in[-1,0]}\left\{\frac{1}{f_0(x)}\left[2\left(1 - \frac{\mu_C}{\mathfrak{B}(x)}\right) + \frac{\mu_C^2 + \sigma_C^2}{\mathfrak{B}^2(x)} - x^2\right]\right\}$. Thus, the final b value is the maximum between the two cases. ■

Remark 2: Actually, the just stated theorem proves existence of an invariant *probability measure*, as defined in section 1.2.1 of [21], and not of a probability density *funtion*. For the sake of simplicity, measures have not been used in this work, even if a formally correct statement of the found results should make use of them.
Given the p.d.f. of an input process modeling a real application, and a control function $\mathfrak{B}(\cdot)$, the stated result allows to check the existence of an invariant p.d.f. for the closed loop system dynamics. This is not all what one may want to check. In fact, theoretically, the sequence $\{f_k\}$ may converge to an invariant p.d.f. only when starting from some states, and it may not converge at all when starting from other states. Furthermore, multiple invariant p.d.f. functions may exist for the system under examination, each one reached starting from a different set of starting states. The following result provides a criterion that allows to assess unicity of the invariant p.d.f., provided that further assumptions on the input process are validated.

Theorem 2: Suppose, in addition to the assumptions made in 1, that the input process p.d.f. $f_C(\cdot)$ is strictly positive in an interval $[a, b]$ such that $a < B_L < B_H < b$, and is null if $c < a$. Then, a unique stationary p.d.f. exists for the s.e. evolution, and the system is ergodic.

Proof: Due to 1, a stationary p.d.f. exists. Then, it is sufficient to apply Proposition 4.2.2 in [21], stating that if a MC is irreducible, and admits an invariant p.d.f., then the invariant p.d.f. is unique and the MC is ergodic. In order to prove irreducibility, we have to show that the expected number of visits to any set A (with non null measure) is positive, given any start state x, which may also be written as: $\forall x \in [0, +\infty[$, $\forall A$ s.t. $\phi(A) > 0$, $\sum_{n=1}^{\infty} P^n(x, A) > 0$, with the transition probability function P given by:

$P(x,A) = \int_A K(y,x)dy$. We will prove that, starting from any state x, there exist a sequence of expanding ranges $\{R_n \equiv [a_n, b_n]\}$ of y values where $K^n(y,x)$ is certainly strictly positive, and that this sequence converges to the entire state space, for $n \to \infty$. In fact, $K(y,x) = f_C\left[\mathfrak{B}(x)(1+y-s(x))\right]\mathfrak{B}(x)$, and $\mathfrak{B}(x) \leq B_H$ implies $\mathfrak{B}(x)(1+y-s(x)) \leq B_H(1+y-s(x)) \leq b$ which is implied by any $y \leq \frac{b}{B_H} + s(x) - 1 > s(x)$. Furthermore, $\mathfrak{B}(x) \geq B_L$ implies $\mathfrak{B}(x)(1+y-s(x)) \geq B_L(1+y-s(x)) \geq a$ which is implied by any $y \geq \frac{a}{B_L} + s(x) - 1 < s(x)$. This means that $a_1 = s(x) - (1 - \frac{a}{B_L})$, $b_1 = s(x) + (\frac{b}{B_H} - 1)]$; $a_{n+1} = s(a_n) - (1 - \frac{a}{B_L})$, $b_{n+1} = s(b_n) + (\frac{b}{B_H} - 1)$, and the upper bound of each set grows of a value $(\frac{b}{B_H} - 1) > 0$ for each set in the sequence. Proof if concluded by observing that the region $\{\varepsilon < -(1 - \frac{a}{B_L})\}$ is never reachable, in any number of steps, thus it is sufficient to exclude it from the state space, and also from the possible x values, in the proof. Thus, on the remaining range $[-(1 - \frac{a}{B_L}), +\infty[$, the MC is completely reachable, thus it is irreducible. ■

Using the above criteria it is immediate to show the following result.

Corollary 1: The SDB and the cost optimal controllers introduced above guarantee existence and uniqueness of an invariant probability function. Furthermore, the resulting $\{\mathcal{E}_k\}$ process is ergodic.

Concerning the invariant-based control, a degree of freedom has been left on the choice of the bandwidth value, which may be any in the range dictated by Equation 5, whenever the current s.e. is inside the invariant. Thus, the just cited stability criteria may be applied once any continuous function has been chosen, with the constraint of connecting continuously to the recovery policy of Equation 6.

Remark 3: Conditions stated in theorem 2 are overly conservative. They can be relaxed still preserving irreducibility of the MC and, eventually, unicity of the invariant distribution. This topic needs further investigation.

C. Properties at the stochastic equilibrium

Proposition 4: Consider the stochastic system defined by Equation 4 evolving under the control of a generic function $\mathfrak{B}(\cdot)$ upper-bounded by $B_H \leq 1$, and for which the job execution times are i.i.d. stochastic variables. Suppose an invariant p.d.f. f exists for the system, and define the expectation operator E_f as $E_f[g(X)] \triangleq \int g(x)f(x)dx$. If both $\mu_f \triangleq E_f[\mathcal{E}] = \int xf(x)dx$ and $E\left[\frac{1}{\mathfrak{B}(\mathcal{E}_k)}\right]$ are finite, then $B_H \geq \mu_C$.

Proof: Equation 4 implies $E\left[\mathcal{E}_{k+1}\right] = E\left[s(\mathcal{E}_k)\right] + E\left[\frac{C_k}{\mathfrak{B}(\mathcal{E}_k)}\right] - 1 = E\left[s(\mathcal{E}_k)\right] + E\left[C_k\right]E\left[\frac{1}{\mathfrak{B}(\mathcal{E}_k)}\right] - 1$, where the last equality holds because of the stochastic independence between C_k, the next job execution time, and $\mathcal{E}_k$, the current job s.e. value. Under the theorem assumptions, and considering a sufficiently large k for which the system has reached the stochastic equilibrium, the p.d.f. of $\mathcal{E}_{k+1}$ must be the same as the p.d.f. of $\mathcal{E}_k$, leading to $\mu_f = E_f\left[s(\mathcal{E}_k)\right] + \mu_C E_f\left[\frac{1}{\mathfrak{B}(\mathcal{E}_k)}\right] - 1 \geq \mu_f + \frac{\mu_C}{B_H} - 1$, from which the stated proposition easily follows. ■

Proposition 5: Given assumptions of Proposition 4, if $\forall \varepsilon \leq 0$, $\mathfrak{B}(\varepsilon) = B_L$, and f leads to a positive probability of negative s.e. $p \triangleq \int_{-1}^{0} f(x)dx > 0$, then the following conditions hold

$$\frac{p}{B_L} + \frac{1-p}{B_H} < \frac{1}{\mu_C}, B_H > \mu_C, p < \frac{B_L}{\mu_C} \quad (10)$$

Proof: Equation 4 implies: $E\left[\mathcal{E}_{k+1}\right] = E\left[s(\mathcal{E}_k)\right] + E\left[\frac{C_k}{\mathfrak{B}(\mathcal{E}_k)}\right] - 1 = E\left[s(\mathcal{E}_k)\right] + E\left[C_k\right]E\left[\frac{1}{\mathfrak{B}(\mathcal{E}_k)}\right] - 1 = \int_0^{+\infty} xf(x)dx + \mu_C\left[\int_{-1}^{0}\frac{1}{B_L}f(x)dx + \int_0^{+\infty}\frac{1}{\mathfrak{B}(x)}f(x)dx\right] - 1 \geq \int_0^{+\infty} xf(x)dx + \frac{\mu_C}{B_L}p + \frac{\mu_C}{B_H}(1-p) - 1$. On the other hand, at the stochastic equilibrium, $E\left[\mathcal{E}_{k+1}\right] = \int_{-1}^{0} xf(x)dx + \int_0^{+\infty} xf(x)dx$ holds, thus it follows

$$\int_{-1}^{0} xf(x)dx \geq \frac{\mu_C}{B_L}p + \frac{\mu_C}{B_H}(1-p) - 1.$$

Now, from the last theorem condition $\int_{-1}^{0} f(x)dx > 0$, immediately follows $\int_{-1}^{0} xf(x)dx < 0$, directly leading to the first proposition assertion. Furthermore, it is possible to write: $0 > \frac{\mu_C}{B_L}p + \frac{\mu_C}{B_H}(1-p) - 1 \geq \frac{\mu_C}{B_H}p + \frac{\mu_C}{B_H}(1-p) - 1 = \frac{\mu_C}{B_H} - 1$, leading to the second assertion. Finally, rewriting this as $1 - \frac{\mu_C}{B_L}p > \frac{\mu_C}{B_H}(1-p) \geq 0$, the third and last assertions are verified. ■

Note 1: The just stated proposition gives guidelines on the choice of the B_H and B_L parameters, within the control law. In fact, first relation in Equation 10 dictates the upper bound to the probability of negative s.e. at the stochastic equilibruim, once B_L, B_H and μ_C are assigned.

V. Experimental results

In this section we report experimental results gathered on a real Linux system. Experiments are based on a modified version of the Linux scheduler implementing a resource reservation policy, based on a slightly modified version of the Constant Bandwidth Server algorithm found in [22]. Implementation of such QoS support into the Linux kernel has mainly been carried on as part of the OCERA project, a EU funded project. The base modifications to the original Linux kernel are based on a loadable module that, once activated into the kernel, intercepts scheduling-related events allowing the execution of further computations. Further architectural details can be found in [14].

In order to prove effectiveness of the proposed approach in controlling the QoS levels experienced by a user, we focused on the context of multimedia applications, and applied our feedback based scheduling mechanism to the *Xine*[1] MPEG video player. The original application has been modified in order to allow us to apply the feedback based QoS control only on the video decoding thread, which is the most critical part of the player from a scheduling standpoint, due to the potentially high dynamic variations on the computation demand. All experiments shown in this

[1]More information can be found at the URL: http://xinehq.de.

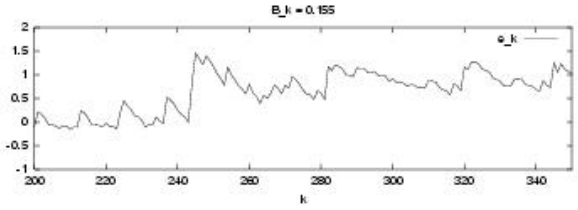

Fig. 2. Scheduling error experienced with a bandwidth statically fixed to $b_k = 0,155$.

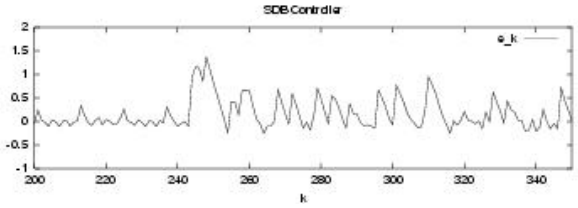

Fig. 3. Scheduling error experienced under the action of a SDB controller with a saturation value of 18%.

section have been done using an AMD Athlon(tm) XP 1800+ based platform, running the Linux OS with kernel 2.4.18.

In the first experiment, we highlight the poor quality that is achieved by allocating a constant bandwidth to the video decoder thread. In fact, for the execution of a movie that requires an average CPU bandwidth of about 15,5%, a static allocation of a bandwidth equal to 15.5% results in unacceptable occasional degradations of the experienced QoS levels during the play, due to temporary increases in the computation demand of the decoder caused by too quickly moving scenes. This is highlighted in Figure , where a time-period has been shown where the scheduling error experiences high positive deviations.

On the other hand, with a static bandwidth allocation of about 18%, the decoder behaves much better during the play, but the decoding thread most times uses a bandwidth that is much higher than strictly required, resulting in an unneeded steal of computation power to other possible applications. This is reflected in a scheduling error most times being negative and high in absolute value (not shown for the sake of brevity).

We activated feedback based scheduling of the decoding thread, when playing the same movie. The controller we used embedded a predictor based on 3 independent moving averages, each based on 4 samples, to predict the next execution time, and a dead-beat controller (with $\varepsilon^* = 0$), with a saturation value of $B_H = 18\%$. This resulted in a movie played substantially as fluidly as with the static allocation of 18%, but an average allocation of the bandwidth of 16% during play. Figure 3 shows the evolution of the s.e. under the action of the feedback based controller, causing the scheduling error to quickly recover from large deviations, and its evolution remaining constrained in a more strict interval.

REFERENCES

[1] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[2] C. W. Mercer, S. Savage, and H. Tokuda, "Processor capacity reserves for multimedia operating systems," Tech. Rep. CMU-CS-93-157, Carnegie Mellon University, Pittsburg, May 1993.

[3] K. Jeffay, F. SMith, A. Moorthy, and J. Anderson, "Proportional share scheduling of operating system services for real-time applications," in *IEEE Real Time System Symposium*, (Madrid, Spain), December 1998.

[4] F. J. Corbato, M. Merwin-Dagget, and R. C. Daley, "An experimental time-sharing system," in *Proceedings of the AFIPS Joint Computer Conference*, May 1962.

[5] T. Nakajima, "Resource reservation for adaptive qos mapping in real-time mach," in *Sixth International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, April 1998.

[6] J. Regehr and J. A. Stankovic, "Augmented CPU Reservations: Towards predictable execution on general-purpose operating systems," in *Proceedings of the IEEE Real-Time Technology and Applications Symposium (RTAS 2001)*, (Taipei, Taiwan), May 2001.

[7] B. Li and K. Nahrstedt, "A control theoretical model for quality of service adaptations," in *Proceedings of Sixth International Workshop on Quality of Service*, 1998.

[8] C. Lu, J. A. Stankovic, T. F. Abdelzaher, G. Tao, S. H. Son, and M. Marley, "Performance specifications and metrics for adaptive real-time systems," in *Proceedings of the 21th IEEE Real-Time Systems Symposium*, (Orlando, FL), December 2000.

[9] D. Steere, A. Goel, J. Gruenberg, D. McNamee, C. Pu, and J. Walpole, "A feedback-driven proportion allocator for real-rate scheduling," in *Proceedings of the Third usenix-osdi*, pub-usenix, feb 1999.

[10] L. Abeni and G. Buttazzo, "Adaptive bandwidth reservation for multimedia computing," in *Proceedings of the IEEE Real Time Computing Systems and Applications*, (Hong Kong), December 1999.

[11] L. Abeni, L. Palopoli, G. Lipari, and J. Walpole, "Analysis of a reservation-based feedback scheduler," in *Proc. of the Real-Time Systems Symposium*, (Austin, Texas), November 2002.

[12] L. Palopoli, L. Abeni, and G. Lipari, "On the application of hybrid control to cpu reservations," in *Hybrid systems Computation and Control (HSCC03)*, (Prague), april 2003.

[13] L. Palopoli, T. Cucinotta, and A. Bicchi, "Quality of service control in soft real-time applications," in *Conference on Decision and Control (CDC)*, 2003.

[14] L. Abeni, T. Cucinotta, G. Lipari, L. Marzario, and L. Palopoli, "Adaptive reservations in a linux environment," in *To appear in Proceedings of RTAS 2004*, 2004.

[15] R. Rajkumar, K. Juvva, A. Molano, and S. Oikawa, "Resource kernels: A resource-centric approach to real-time and multimedia systems," in *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking*, January 1998.

[16] L. Abeni and G. Buttazzo, "Integrating multimedia applications in hard real-time systems," in *Proceedings of the IEEE Real-Time Systems Symposium*, (Madrid, Spain), December 1998.

[17] G.Lipari and S. Baruah, "Greedy reclaimation of unused bandwidth in constant bandwidth servers," in *IEEE Proceedings of the 12th Euromicro Conference on Real-Time Systems*, (Stokholm, Sweden), June 2000.

[18] D. Reed and R. F. (eds.), "Nemesis, the kernel – overview," May 1997.

[19] G. Lipari, G. Buttazzo, and L. Abeni, "A bandwidth reservation algorithm for multi-application systems," in *IEEE Real Time Computing Systems and Applications*, (Hiroshima, Japan), October 1998.

[20] L. Palopoli and T. Cucinotta, "QoS control in reservation-based scheduling," tech. rep., Scuola Superiore S. Anna, 2003.

[21] J. Lasserre and O. Hernandez-Lerna, *Markov chains and invariant probabilities*, vol. Progress in mathematics: Volume 211. Birkhaeuser Verlag, second ed., 2003.

[22] L. Abeni, "Server mechanisms for multimedia applications," Tech. Rep. RETIS TR98-01, Scuola Superiore S. Anna, 1998.